

The Beginnings at MIT

Two key elements were necessary to implement time-shared, interactive computing: interfaces with communications facilities, and a machine design that supported interrupts, memory protection, and a large fast-access external store. Each of these elements was feasible in 1959, as demonstrated by George R. Stibitz in 1940 and by the SAGE development in 1957.

The first public demonstration of remote operation of a digital computer and of computer-telephone communications was conducted as part of the 1940 annual meeting of the Mathematical Association of America. Problems were entered into a teletypewriter located in McNutt Hall on the Dartmouth campus in Hanover, N.H. They were then transmitted via standard Bell System telecommunications facilities to the Bell Telephone Laboratories in New York City, where they were solved on-line by the BTL Model 1 Complex Number Calculator. The results were immediately returned on-line via the same telecommunications link to the teletypewriter located on the Dartmouth campus, 250 miles away.

Stibitz, who was behind both the demonstration and the digital computer used in it, was then a mathematician on the technical staff of the Bell Telephone Laboratories. (Now he is professor emeritus of physiology at the Dartmouth Medical School.)

Stibitz describes the ability to do remote computing as a natural and integral part of the design of his system. It was natural and “no big thing” to demonstrate this device at Dartmouth by leaving the computer where it was and operating it remotely via a teletypewriter link, rather than going to all the trouble and expense of moving the machine itself from New York to Hanover. After all, teletypewriters in other departments in BTL had already tapped into the system on occasion to get their computations done. The length of the teletype line, whether it was 25 feet or 250 miles, involved no major change in operational techniques or concept.²⁹

In their 1957 paper, “SAGE — A Data Processing System for Air Defense” (*Proc. EJCC* © 1957 IRE (now IEEE)), R.R. Everett, C.A. Zraket, and H.D. Bennington²⁰ described the working of the SAGE system and not only used the term “time-sharing” but also provided their own description:

The central computer performs air-defense processing in the following manner (see figure 1 [repro-

duced on the facing page]). The buffer storage tables, the system-status data, and the system computer program are organized in hundreds of blocks — each block containing from 25 to 4000 computer words. A short sequence-control program in the central computer’s core memory transfers appropriate program and data blocks into core memory, initiates processing, and then returns appropriate table blocks (but never programs) back to the drum. To take advantage of the in-out break feature, operation of each air-defense routine is closely coordinated with operation of the sequence-control program so that programs and data are transferred during data processing.

By *time-sharing* [emphasis added] the central computer, each of the air-defense routines is operated at least once every minute — many are operated every several seconds. One interesting feature is that the frequency of program operation is locked with real time rather than allowed to vary as a function of load; during light load conditions the sequence-control program will often “mark time” until the real-time clock indicates that the next operation should be repeated. Such synchronization with real time simplifies many of the control and input-output functions without causing any degradation in system performance.

Both of these demonstrations showed the feasibility of the concept, but it took an environment (at MIT) and a brilliant mind (John McCarthy) to put two and two together into a far-reaching system.

Digital computation* began at MIT in 1947 with the Whirlwind I computer. Whirlwind was four or five times faster than its predecessors by virtue of its excellent electronic design and parallel operation. By 1953, core memory, developed by Jay Forrester and his associates at MIT, yielded another factor of two. The standards of speed and reliability set at that time have now come to be expected of modern computing equipment.

While Whirlwind satisfied the needs of many MIT users, the pressures of special requirements led to the acquisition of other computers as well. In 1955 the Instrumentation

* Of course in the field of analog computation, Vannevar Bush had led the world with the development of the Differential Analyzer.

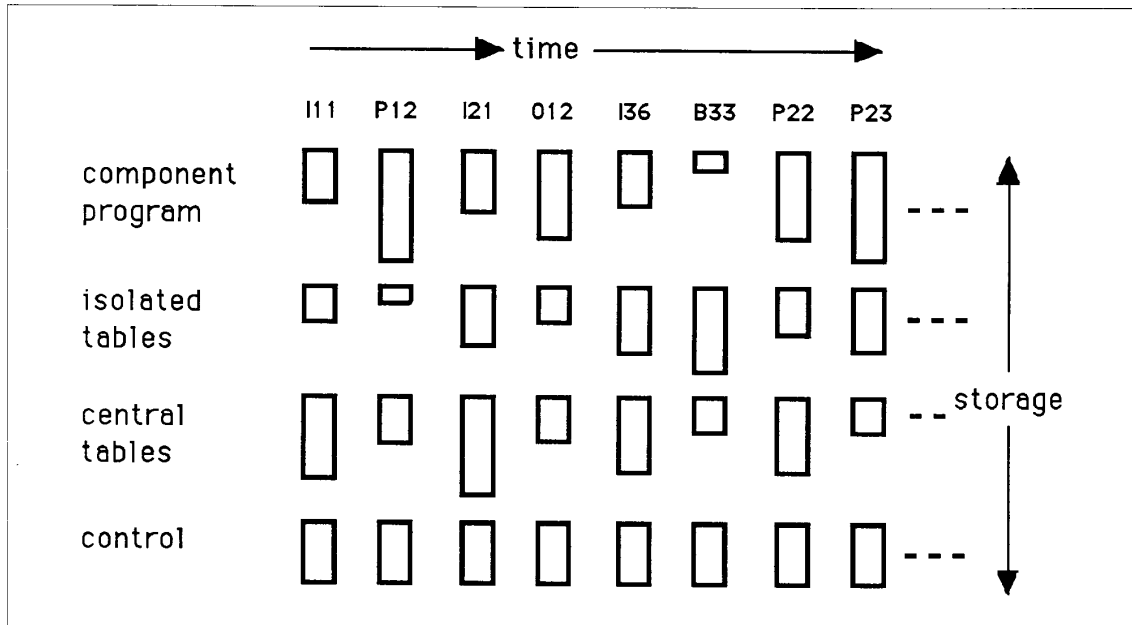


Figure 1. Dynamic program operation.²⁰ Reproduced with permission (© 1957 IRE (now IEEE)).

Laboratory obtained an IBM 650 computer, and in 1956 the Naval Supersonic Laboratory installed a Bendix G15 computer for processing wind tunnel data.

In 1956, IBM provided an IBM 704 to be located at MIT with one shift per day for the use, without charge, of educational and research projects at MIT, and a similar amount of time for the use of universities and colleges in the New England area. The Computation Center, under the direction of Philip M. Morse, was formed for the administration and efficient use of this facility. Subsequently, the TX-0 computer was loaned to the Electrical Engineering Department by Lincoln Laboratory.

The need for computer capacity at MIT continually increased. In 1960 the Computation Center IBM 704 was replaced by the more powerful IBM 709 computer, and several smaller computers also were installed by various MIT groups. Even more ambitious expansion was scheduled for the immediate future.

Reminiscences on the History of Time-Sharing

John McCarthy

At the time of our interviews with the CTSS and Project MAC pioneers, we invited John McCarthy to join the group which provided this record following the 25th anniversary celebrations of the founding of the Laboratory of Computer Science. McCarthy was unable to extend his stay in the Cam-

bridge area but graciously wrote his own memories of this era.

I remember thinking about time-sharing at the time of my first contact with computers and being surprised that this was not the goal of IBM and all the other manufacturers and users of computers. This might have been around 1955.

By time-sharing, I meant an operating system that permits each user of a computer to behave as though he were in sole control of a computer, not necessarily identical with the machine on which the operating system is running. Christopher Strachey may well have been correct in saying in his letter to Donald Knuth* that the term was already in use for time-sharing among programs written to run together. This idea had already been used in the SAGE system. I do not know how this kind of time-sharing was implemented in SAGE. Did each program have to be sure to return to an input polling program or were there interrupts? Who invented interrupts anyway?** I thought of them, but I do not believe I mentioned the idea to anyone before I heard of them from other sources.

My first attempts to do something about time-sharing were in the fall of 1957 when I came to the MIT Computation Center on a Sloan Foundation fellowship from Dartmouth College. It was immediately clear to me that time-sharing the IBM 704 would require some kind of interrupt system. I was very shy of proposing hardware modifications,

* See Strachey's response to Knuth in the previous section.

** Alan Scherr reported that the patents on interrupt-driven I/O are held by three IBM researchers.

John McCarthy's 1959 memorandum

To: Professor P.M. Morse
From: John McCarthy
Subject: A Time-sharing Operator
Program for our
Projected IBM 709
Date: January 1, 1959

Introduction

This memorandum is based on the assumption that MIT will be given a transistorized IBM 709 about July 1960. I want to propose an operating system for it that will substantially reduce the time required to get a problem solved on the machine. Any guess as to how much of a reduction would be achieved is just a guess, but a factor of five seems conservative. A smaller factor of improvement in the amount of machine time used would also be achieved.

The proposal requires a complete revision in the way the

The format of the original memo was changed for reproduction here.

machine is used, will require a long period of preparation, the development of some new equipment, and a great deal of cooperation and even collaboration from IBM. Therefore, if the proposal is to be considered seriously, it should be considered immediately. I think the proposal points to the way all computers will be operated in the future, and we have a chance to pioneer a big step forward in the way computers are used. The ideas expressed in the following sections are not especially new, but they have formerly been considered impractical with the computers previously available. They are not easy for computer designers to develop independently since they involve programming system design much more than machine design.

A quick service computer

Computers were originally developed with the idea that programs would be written to solve general

classes of problems and that after an initial period most of the computer time would be spent in running these standard programs with new sets of data. This view completely underestimated the variety of uses to which computers would be put. The actual situation is much closer to the opposite extreme, wherein each user of the machine has to write his own program and that once this program is debugged, one run solves the problem. This means that the time required to solve the problem consists mainly of time required to debug the program. This time is substantially reduced by the use of better programming languages such as Fortran, Lisp (the language the Artificial Intelligence Group is developing for symbolic manipulations) and COMIT (Yngve's language). However, a further large reduction can be achieved by reducing the response time of the computation center.

especially as I did not understand electronics well enough to read the logic diagrams. Therefore, I proposed the minimal hardware modification I could think of. This involved installing a relay so that the 704 could be put into trapping mode by an external signal. It was also proposed to connect the sense switches on the console in parallel with relays that could be operated by a Flexowriter.

When the machine went into trapping mode, an interrupt to a fixed location would occur the next time the machine attempted to execute a jump instruction (then called a transfer). The interrupt would occur when the Flexowriter had set up a character in a relay buffer. The interrupt program would then read the character from the sense switches into a buffer, test whether the buffer was full, and if not return to the interrupted program. If the buffer was full, the program would store the current program on the drum and read in a program to deal with the buffer.

It was agreed (I think I talked to Dean Arden only) to install the equipment, and I believe that permission was obtained from IBM to modify the computer. The connector to be installed in the computer was obtained.

However, at this time we heard about the "real-time package" for the IBM 704. This RPQ (Request for Price Quotation was IBM jargon for a modification to the computer whose price was not guaranteed), which rented for \$2,500 per month, had been developed at the request of Boeing for the purpose of allowing the 704 to accept information from a wind tunnel. Some element of ordinary time-

sharing would have been involved, but we did not seek contact with Boeing. Anyway, it was agreed that the real-time package, which involved the possibility of interrupting after any instruction, would be much better than merely putting the machine in trapping mode. Therefore, we undertook to beg IBM for the real-time package. IBM's initial reaction was favorable, but nevertheless it took a long time to get the real-time package — perhaps a year, perhaps two.

It was then agreed that someone, perhaps Arnold Siegel,* would design the hardware to connect one Flexowriter to the computer, and later an installation with three would be designed. Siegel designed and built the equipment, the operating system was suitably modified (I do not remember by whom), and a demonstration of on-line Lisp was held for a meeting of the MIT Industrial Affiliates. This demonstration, which I planned and carried out, had the audience in a fourth-floor lecture room and me in the computer room and a rented closed-circuit TV system. Steve Russell, who worked for me, organized the practical details, including a rehearsal. This demonstration was called time-stealing, and was regarded as a mere prelude to proper time-sharing. It involved a fixed program in the bottom of memory that collected characters from the Flexowriter in a buffer while an ordinary batch job was running. It was only after each job was run that a job that would deal with the

* Siegel had engineered a Flexowriter input device on Whirlwind for Doug Ross' Data Reduction Project in 1956.

The response time of the MIT Computation Center to a performance request presently varies from 3 hours to 36 hours depending on the state of the machine, the efficiency of the operator, and the backlog of work. We propose by time-sharing, to reduce this response time to the order of 1 second for certain purposes. Let us first consider how the proposed system looks to the user before we consider how it is to be achieved.

Suppose the average program to be debugged consists of 500 instructions plus standard subroutines and that the time required under the present system for an average debugging run is 3 minutes. This is time enough to execute 7,000,000 704 instructions or to execute each instruction in the program 14,000 times.

Most of the errors in programs could be found by single-stepping or multiple-stepping the program as used to be done. If the program is debugged in this way, the program

will usually execute each instruction not more than 10 times, 1/1400 as many executions as at present. Of course, because of slow human reactions the old system was even more wasteful of computer time than the present one. Where, however, does all the computer time go?

At present most of the computer time is spent in conversion (SAP-binary, decimal-binary, binary-decimal, binary-octal) and in writing tape and reading tape and cards.

Why is so much time spent in conversion and input output?

1. Every trial run requires a fresh set of conversions.

2. Because of the slow response time of the system it is necessary to take large dumps for fear of not being able to find the error. The large dumps are mainly unread, but nevertheless, they are necessary. To see why this is so, consider the behavior of a programmer reading

his dump. He looks at where the program stopped. Then he looks at the registers containing the partial results so far computed. This suggests looking at a certain point in the program. The programmer may find his mistake after looking at not more than 20 registers out of say 1000 dumped, but to have predicted which 20 would have been impossible in advance and to have reduced the 1000 substantially would have required cleverness as subject to error as his program. The programmer could have taken a run to get the first register looked at, then another run for the second, etc., but this would have required 60 hours at least of elapsed time to find the bug according to our assumptions and a large amount of computer time for repeated loading and re-runnings. The response time of the sheet paper containing the dump for any register is only a few seconds, which is OK except that one dump does not usually contain

(continued on the following page)

characters typed in would be read in from the drum. This job would do what it could until more input was wanted and would then let the operating system go back to the batch stream. This worked for the demonstration, because at certain hours the MIT Computation Center operated a batch stream with a time limit of one minute on any job.

Around the time of this demonstration, Herbert Teager came to MIT as an assistant professor of electrical engineering and expressed interest in the time-sharing project. Some of the ideas of time-sharing overlapped some ideas he had while on his previous job, but I do not remember what they were. Philip Morse, the director of the Computation Center, asked me if I was agreeable to turning over the time-sharing project to Teager, since artificial intelligence was my main interest. I agreed to this, and Teager undertook to design the three-Flexowriter system. I'm not sure it was ever completed. There was a proposal for support for time-sharing submitted to the National Science Foundation, and money was obtained. I do not remember whether this preceded Teager, and I do not remember what part I had in preparing it or whether he did it after he came. This should be an important document, because it contains that year's conception of and rationale for time-sharing.*

Besides that, IBM was persuaded to make substantial modifications to the IBM 709 to be installed at the MIT

Computation Center. These included memory protection and relocation, and an additional 32,768 words of memory for the time-sharing system. Teager was the main specifier of these modifications. I remember my surprise when IBM agreed to his proposals. I had supposed that relocation and memory protection would greatly slow the addressing of the computer, but this turned out not to be the case.

Teager's plans for time-sharing were ambitious and, it seemed to many of us, vague. Therefore, Fernando Corbató undertook an "interim" solution using some of the support that had been obtained from NSF for time-sharing work. This system was demonstrated some time in 1961, but it was not put into regular operation. That was not really possible until the ARPA support for Project MAC permitted buying a separate IBM 7090.

Around 1960 I began to consult at BBN on artificial intelligence and explained my ideas about time-sharing to Ed Fredkin and J.C.R. Licklider. Fredkin, to my surprise, proposed that time-sharing was feasible on the PDP-1 computer. This was DEC's first computer, and BBN had the prototype. Fredkin designed the architecture of an interrupt system and designed a control system for the drum to permit it to be used in a very efficient swapping mode. He convinced Ben Gurley, the chief engineer for DEC, to build this equipment. It was planned to ask NIH (National Institutes for Health) for support, because of potential medical applications of time-sharing computers, but before the proposal could even be written, Fredkin left BBN. I took technical

* Regrettably, we have been unable to locate this proposal, but a later report on the work is included in this issue.

information enough to get the entire program correct.

Suppose that the programmer has a keyboard at the computer and is equipped with a substantial improvement on the TX-0 interrogation and intervention program (UT3). (The improvements are in the direction of expressing input and output in a good programming language). Then he can try his program, interrogate individual pieces of data or program to find an error, make a change in the source language and try again.

If he can write the program in source language directly into the computer and have it checked as he writes it, he can save additional time. The ability to check out a program immediately after writing it saves still more time by using the fresh memory of the programmer. I think a factor of 5 can be gained in the speed of getting programs written and working over present practice, if the above mentioned facilities are provided. There is

another way of using these facilities which was discussed by S. Ulam a couple of years ago. This is to use the computer for trial and error procedures where the error correction is performed by a human adjusting the parameters.

The only way quick response can be provided at a bearable cost is by time-sharing. That is, the computer must attend to other customers while one customer is reacting to some output.

The problem of a time-sharing operation system

I have not seen any comprehensive written treatment of the time-sharing problem and have not discussed the problem. This treatment is certainly incomplete and is somewhat off-the-cuff. The equipment required for time-sharing is the following.

- a. Interrogation and display devices (Flexowriters are

possible but there may be something better and cheaper);

- b. An interrupt feature on the computer; we'll have it;
- c. An exchange to mediate between the computer and the external devices. This is the most substantial engineering problem, but IBM may have solved it.

In general the equipment required for time-sharing is well understood, is being developed for various advanced computers, e.g., Stretch, TX-0, Metrovich 1010, Edsac 3. I would not be surprised if almost all of it is available with the transistorized IBM 709. However, the time-sharing has been worked out mainly in connection with real-time devices. The programs sharing the computer during any run are assumed to occupy prescribed areas of storage, debugged already, and to have been written together as a system. We shall have to deal with a continuously

charge of the project as a one-day-a-week consultant, and Sheldon Boilen was hired to do the programming. I redesigned the memory extension system proposed by DEC and persuaded them to build the modified system instead of the two systems they were offering, but fortunately had not built. I also supervised Boilen.

Shortly after this project was undertaken, DEC decided to give a PDP-1 to the MIT Electrical Engineering Department. Under the leadership of Jack Dennis,* this computer was installed in the same room as the TX-0 experimental transistorized computer that had been retired from Lincoln Laboratory when the TX-2 was built. Dennis and his students undertook to make a time-sharing system for it. The equipment was similar, but they were given less memory than the BBN project had. There was not much collaboration.

My recollection is that the BBN project was finished first in the summer of 1962, but perhaps Corbató remembers earlier demonstrations of CTSS.** I left for Stanford in the fall of 1962. I had not seen CTSS, and I believe I had not seen Dennis' system operate either. BBN did not operate the first system and did not even fix the bugs. They had few computer users and were content to continue the system whereby users signed up for the whole computer. They did

undertake a much larger follow-on project involving a time-shared PDP-1 that was installed in Massachusetts General Hospital, where it was not a success. The computer was inadequate, there were hardware and software bugs, and there was a lack of application programs — but mainly the project was premature.

At the same time that CTSS, the BBN system, and the Electrical Engineering Department systems were being developed, MIT had started to plan for a next-generation computer system. The management of MIT evidently started this as an ordinary university planning exercise and appointed a high-level committee consisting of Philip Morse, Albert Hill, and Robert Fano to supervise the effort. However, the actual computer scientists were persuaded that a revolution in the way computers were used was called for. The lower level committee was chaired by Teager, but after his ideas clashed with those of everyone else, the committee was reconstituted with me as chairman. The disagreement centered around how ambitious to be and whether to go for an interim solution. Teager wanted to be very ambitious, but the rest of us thought his ideas were vague. He wanted MIT to acquire an IBM 7030 (Stretch) computer as an interim solution. As it turned out, acquiring a Stretch would have been a good idea.

Our second report to MIT proposed that MIT send out a request for proposals to computer manufacturers. On the basis of the responses, we would then ask the government for the money. The RFP was written, but MIT stalled,

* Earl Pugh of the Electrical Systems Laboratory, which was given the responsibility for the PDP-1, did the original installation and operation of the PDP-1. [Note added by Doug Ross during review.]

** See Time Line, pp. 14-15.

changing population of programs, most of which are erroneous.

The major problems connected with time-sharing during program development seem to be as follows:

1. Allocating memory automatically between the programs. This requires that programs be assembled in a relocatable form and have a preface that enables the operator program to organize the program, its data, and its use of common subroutines.

2. Recovery from stops and loops. The best solutions to these problems require:

- a. Changing the stop instruction to trap instructions. This is a minor modification to the machine. (At least it will be minor for the 704.)
- b. Providing a real-time alarm clock as an external device.

3. Preventing a bad program from destroying other programs. This could be solved fairly readily

with a memory range trap which might not be a feasible modification. Without it, there are programming solutions which are less satisfactory but should be good enough. These include:

- a. Translations can be written so that the programs they produce cannot get outside their assigned storage areas. A very minor modification would do this to Fortran.
- b. Checksums can be used for machine language programs.
- c. Programming techniques can be encouraged which make destruction of other programs unlikely.
- d. There is an excessive tendency to worry about this point. The risk can be brought down to the present risk of having a program ruined by operator or machine error.

Summary

- a. We may be able to make a major advance in the art of using a computer by adopting a time-sharing operator program for our hoped-for 709.
- b. Such a system will require a lot of advance preparation starting right away.
- c. Experiments with using the Flexowriter connection to the real-time package on the 704 will help, but we cannot wait for the results if we want a time-sharing operator program in July 1960.
- d. The cooperation of IBM is very important, but it should be to their advantage to develop this new way of using a computer.
- e. I think other people at MIT than the Computation Center staff can be interested in the systems and other engineering problems involved.

perhaps for two reasons. The first reason was that our initial cost estimates were very large for reasons of conservatism. Second, IBM asked MIT to wait, saying that they would make a proposal to meet MIT's needs at little or no cost. Unfortunately, the System/360 design took longer than IBM management expected, and along about that time, relations between MIT and IBM became very strained because of the patent lawsuit about the invention of magnetic core memory.

As part of the stall, president Stratton proposed a new study with a more thorough market survey to establish the demand for time-sharing among MIT computer users. I regarded this as analogous to trying to establish the need for steam shovels by market surveys among ditch diggers, and I did not want to do it. About this time George Forsythe invited me to come back to Stanford with the intention of building a Computer Science Department, and I was happy to return to California.

In all this, there was not much publication. I wrote a memo to Morse dated January 1, 1959, proposing that we time-share our expected "transistorized IBM 709." It has been suggested that the date was in error and should have been 1960. I do not remember now, but I believe that if the memo had been written at the end of 1959, it would have referred to the 7090, because that name was by then current. In that memo I said the idea of time-sharing was not especially new. I do not know why I said that, except that I did

not want to bother to distinguish it from what was done in the SAGE system with which I was not very familiar.

Most of my argumentation for time-sharing was oral, and when I complained about Fano and Corbató crediting Strachey with time-sharing in their 1966 *Scientific American* article, Corbató was surprised to find my 1959 memo in the files. Their correction in *Scientific American* was incorrect, because they supposed that Strachey and I had developed the idea independently, whereas giving each user continuous access to the machine was not Strachey's idea at all. In fact, he did not even like the idea when he heard about it.

Teager and I prepared a joint abstract for an ACM meeting shortly after he arrived, and I gave a lecture in an MIT series called *Computers and the World of the Future*.³⁴ In this lecture I referred to Strachey's paper "Time-sharing in Large Fast Computers"⁴³ given at the 1959 IFIP Congress in Paris. I had read the paper carelessly, and supposed he meant the same thing as I did. As he subsequently pointed out, he meant something quite different that did not involve a large number of users, each behaving as though he had a machine to himself. As I recall, he mainly referred to fixed programs, some of which were compute bound and some input-output bound. He did mention debugging as one of the time-shared activities, but I believe his concept involved one person debugging while the other jobs were of the conventional sort.

My 1959 memo advertised that users generally would get the advantage of on-line debugging. However, it said nothing

Time-Sharing at MIT

ing about how many terminals would be required and where they would be located. I believe I imagined them to be numerous and in the users' offices, but I cannot be sure. Referring to an "exchange" suggests that I had in mind many terminals. I cannot now imagine what the effect was on the reader of my failure to be explicit about this point. I'm afraid I was trying to minimize the difficulty of the project.

The major technical error of my 1959 ideas was an underestimation of the computer capacity required for time-sharing. I still do not understand where all the computer time goes in time-sharing installations, and neither does anyone else.

Besides MIT's NSF proposal, there ought to be some letters to IBM and perhaps some IBM internal documents about the proposal, since they put more than a million dollars worth of equipment into it. Gordon Bell discusses DEC's taking up time-sharing in the Bell/Newell book,* but I do not recall that they discuss Ben Gurley's role. Fredkin and perhaps Allan Kotok would know about that.

After I came to Stanford, I organized another PDP-1 time-sharing project. This was the first time-sharing system based on display terminals. It was used until 1969 or 1970 for [Patrick] Suppes' work on computer-aided instruction.

Excerpts from "Man-Computer Symbiosis"²⁸

About the same time McCarthy was writing his memorandum to Morse on the mechanics of time-sharing, J.C.R. Licklider was investigating the concept of man-computer symbiosis, a premonition of man-machine cooperation which is even today not wholly achieved. The following selection from Licklider's "Man-Computer Symbiosis" is reprinted with permission from IRE Transactions on Human Factors in Electronics, March 1960 (© 1960 IRE (now IEEE)).

Man-machine symbiosis is an expected development in cooperative interaction between man and electronic computers. It will involve close coupling between the human and electronic members of the partnership. The main aims are 1) to let computers facilitate formulative thinking as they now facilitate the solution of formulated problems, and 2) to enable man and computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs. In the anticipated symbiotic partnership, men will set the goals, formulate the hypotheses, determine the criteria, and perform the evaluations. Computing machines will do the routinizable work that must be done to prepare the way for insights and decisions in technical and scientific thinking. Preliminary analyses indicate that the symbiotic relationship will perform intellectual operations much more effectively than man alone can perform them. Prerequisites for the achievement of the effective, cooperative association include developments in computer time-sharing, in memory components,

in memory organization, in programming languages, and in input and output equipment.

In one sense of course, any man-made system is intended to help man, to help a man or men outside the system. If we focus upon the human operator(s) within the system, however, we see that, in some areas of technology, a fantastic change has taken place during the last few years. "Mechanical extension" has given way to replacement of men, to automation, and the men who remain are there more to help than to be helped. In some instances, particularly in large computer-centered information and control systems, the human operators are responsible mainly for functions that it proved infeasible to automate. Such systems are not symbiotic systems. They are "semi-automatic" systems, systems that started out to be fully automatic but fell short of the goal.

It seems entirely possible that, in due course, electronic or chemical "machines" will outdo the human brain in most of the functions we now consider exclusively within its province.

It is often said that programming for a computing machine forces one to think clearly, that it disciplines the thought process. If the user can think his problem through in advance, symbiotic association with a computing machine is not necessary.

However, many problems that can be thought through in advance are very difficult to think through in advance. They would be easier to solve, and they could be solved faster, through an intuitively guided trial-and-error procedure in which the computer cooperated, turning up flaws in the reasoning or revealing unexpected turns in the solution. Poincaré anticipated the frustration of an important group of would-be computer users when he said "The question is not 'What is the answer?' The question is 'What is the question?'" One of the main aims of man-computer symbiosis is to bring the computing machine effectively into the formulative parts of technical problems. The other aim is to...bring computing machines effectively into the processes of thinking that must go in "real time," time that moves too fast to permit using computers in conventional ways.

Later Licklider was instrumental in not only initiating the concept of an extensive project which would have similar aims to those expressed in his 1960 paper, but also he was in a position to provide the funding for that work! His train ride with Robert Fano from Hot Springs, Va., to Washington, D.C. (probably aboard the "Crescent") gave them the opportunity to realize that ARPA needs could be met through the capabilities of the MIT group. In our interview with Licklider and Fano we asked them of their recollections of this trip.

Teager's recommendation for an IBM 7030⁴⁸

The two studies that preceded the formalization of the MIT activity, which led first to the development of the Compatible Time-Sharing System (CTSS) and eventually to Proj-

* More likely: C.G. Bell, J.C. Mudge, and J.E. McNamara, *Computer Engineering*, Digital Press, Bedford, Mass., 1978.

ect MAC, have been mentioned both by McCarthy in his recollections and later by the respondents in the interviews. The first study group report, written by Herbert Teager, recommended the acquisition of an IBM 7030 (Stretch), although Teager recognized that he was unable to obtain all the necessary data regarding its capabilities, and the support software was an unknown factor. Retrospectively, Teager's arguments were sound; his choice of the IBM Stretch was unfortunate.

MIT should obtain, within the next two to three years, an ultra large capacity computer, develop time-shared remote input-output facilities complete with display and graphical input capability, and begin an intensive effort to develop advanced, user oriented programming languages for this system. This policy would seem the best possible way for MIT to obtain a research facility to multiply the intellectual effectiveness of the experimental and theoretical research workers and teachers in all fields. The policy would at the same time provide a necessary experimental tool for frontier research in many fields involving physical simulation, information processing, and real-time experimental work, which would otherwise be unfeasible.

The machine should have sufficient capacity and be of sufficiently advanced design so that it would have a useful life of at least six years, before any further change need be contemplated.

Based upon presently published specifications for existing machines, the IBM 7030, Stretch Computer is a suitable candidate for the recommended computer, by virtue of its speed, memory size, and overall capacity and capabilities.

Published data for Stretch, while not including some important, but presently unknown factors such as reliability and mean, error-free running time, does indicate that the commercial version of this machine will come very close to meeting its specifications. Several have been built, and programs are operating on this machine as the last of the "bugs" are being wrung out. Other announced machines are either unsuitable or else are in such an early stage of development that specifications and costs are too vague for an objective appraisal.

The final decision for the central computer should be made on a basis of its installation within a maximum period of perhaps *three years* [emphasis added]; a minimum set of speed, reliability, memory size, and overall capacity requirements; and finally the relative cost for such unit capacity. Presently unknown commercial machines might conceivably prove a better choice if they could meet these criteria, and in any event other manufacturers should be given a chance to bid.

In order to properly prepare for such a machine, it is believed that the machine design and programming specifications would have to be firm within a year at most. Before IBM or any other computer manufacturer can be directly approached at a high level for performance specifications and bids, it is believed that an early policy decision with respect to MIT's intentions is necessary.

The present trend towards additional separate computers would appear an unsatisfactory overall solution to MIT's

problems in this area from a standpoint of both cost and overall performance. Far more serious, however, is the fact that the chance to provide the needed capacity in the form recommended by this Report may well become impossible in the face of the developments and acquisitions of these many divergent groups unless a decision is made very soon.

Making the capacity of a powerful machine more directly available to each user will multiply its effectiveness and eliminate the drawbacks of a large central machine.

Analyses based upon costs and characteristics of existing commercial machines made during the course of this study tend to show that computer capacity is far more economically provided by a single, very large capacity machine, rather than by many separate medium- to small-scale machines. In addition, there are vital research problems scattered in all fields of interest to the Institute which must have vast memory sizes and extremely high processing speeds if they are to be solved within feasible running times. Separate machines which may have either a vast memory or high speed are thus highly inefficient and overly costly both in total time and money for the solution of these problems. Providing a capacity of the type necessary can most economically be achieved by the rental or purchase of the largest, fastest available commercial machine such as the IBM Stretch computer. There is, at present, no other existing machine or combination of machines that can achieve comparable computation costs and rates.

Making the capacity of a powerful machine more directly available to each user will multiply its effectiveness, and eliminate the drawbacks of the use of a large central machine such as have been experienced at MIT. The MIT Computation Center is as well administered as any other comparable facility with a load of similar nature and magnitude. Nearly 400 distinct problems are active at the Center at any given time, and to fairly distribute the available time, amounting to less than half of that requested, among large numbers of anxious users, leads to difficulties. These become even greater as nearly 100 competing users attempt to use the machine on an average day. The net result from the user's viewpoint is excessive paper work to achieve inadequate time grants, long turnaround times between program submission and results, and thus, an overall slowdown in research. There would be little justification for an extremely high capacity central facility which compounded these shortcomings on a larger scale. Fortunately, however, there is strong technical evidence that it need not.

While some of the faculty members are aware of computers and programming developments with a potential application in their field, they are even more aware of the present usage difficulties. Programming for them, even in the so-called advanced languages, tends to be tedious, not easily applicable to their own work, and requiring knowl-

Time-Sharing at MIT

edge of far too many conventions and exceptions. Other obstacles are the uninterrupted hours required to program and debug before finally getting any results, in the face of an overcrowded faculty day, and an overworked faculty. They cannot afford to wait for the presently necessary weeks of elapsed time between problem formulation and machine solution.

The difficulty of administering a large central machine can only be overcome if a time-shared central machine is brought out to many simultaneous users in the form of remote input-output "consoles" which have all the characteristics of a user's own personal computer with respect to access, a known available capacity, and a minimum of usage formalities. Such a console would provide far more capacity and capability than the user could afford for the same cost in a machine of his own. [See Figure 2.] Such capacity, in addition to speed, would allow the user to "program" in languages that are closely allied to that of his field, and in addition, eventually allow him to use pencil and paper should he dislike or be unable to use a keyboard to communicate his problem to the machine. It would also provide graphical display and a permanent record of all work; such capability would go a very long way towards attracting a sizable fraction of the faculty who could use computers to extreme advantage, but who cannot spare the protracted time requirements of present usage, as well as realizing the benefits of computers as a classroom aid.

During evenings, nights and weekends, the system described would have adequate capacity to run very large problems, which might require the total capacity of the central machine, or the work of groups which might not be compatible with or desirable for time-sharing. This would be a minor restriction upon such groups.

Although development programs are already underway for the high capability, low-cost console described, time-sharing techniques and remote input-output stations equipped with typewriters presently exist. A large part of the access problem described can be solved today with very little further development, and thus the entire program does not hinge on the possible gamble that to develop the high-capability languages and facilities might take longer than the indicated three-year time period. To provide the central facility will be costly, as will the programming effort to provide the needed languages, and to a lesser extent, the development and procurement of a large number of high-capability remote facilities. But it would seem the cheapest way of providing the needed capacity, and the only way of providing the necessary, close man-machine interrelationship which is felt vital for research in the years to come.

One of the most common complaints made by users in the course of the survey of all MIT research projects was that it was extremely difficult to find competent programmers, who could translate their desires into machine programs. This is a symptom of a much longer range problem, i.e., non-faculty usage, and calls for a longer range solution.

For any intellectual tool to be used, it must have a response time roughly comparable to that of the person using it; otherwise it may well slow down, rather than speed up, the overall rate of the man and machine. The net result

will be that the user does not use the tool, even though it *could be* of material assistance to him.

It is believed that the relatively small percentage of the MIT faculty who are using computers directly, and the rather limited amount of such usage, is due primarily to the tedious nature of present day programming languages, and the inescapable fact that it requires a minimum of 2-4 weeks to get a moderately complex program written in the best available language to operate... For most senior researchers, this time lag provides a sufficiently strong barrier to keep them from using machines at all.

Much has already been made in this study of the fact that MIT faculty are not directly using computers but are rather working through middlemen, in the form of research staff and graduate student programmers. More will be made of the ineptitude of present languages, which results in many "debugging" and compiling runs before a problem is finally ready for the "production" which was its *raison d'être*. Due to present administration policies and computation loads, most of the total elapsed time in getting to production is spent in the presently *inevitable* [emphasis added] waits of turnaround times between submissions in the succession of reruns.

The net result is to force highly skilled personnel to remain idle for long periods of time, unless sufficient machine time is granted so that they are working on several problems at the same time, but having to constantly "shift mental gears" and rethink programs.

As their efficiency decreases (due to the long turnaround times and small time grants) so does morale, and not having the satisfaction of seeing their work come to early fruition, the best of them leave.

No study was made of this area, but from many personal contacts and discussions it is estimated that no group at MIT can expect to hire and keep good qualified programmers for much longer than about 1½ years. Many groups have suffered almost 100 per cent casualties inside of three years. This is not, it is believed, a question of salaries.

The effect of such a turnover rate, the constant loss of the best and most talented people, and the training of far less capable ones to take their place are highly detrimental and costly to the research work of MIT. This problem too, it is believed, can be alleviated.

In the past, many cogent and pressing arguments have been presented for separate small- to medium-scale computers for use by many of the research groups around Tech. Although a variety of reasons have been given for this form of private enterprise, the basic reasons it is believed are ones of access and possession. A researcher in general regards a computer as a tool (ranging in power from an intelligent assistant to a desk calculator, depending upon his inclination and familiarity) and, as with all other tools, feels that its effectiveness is measured in part by his degree of access to it, and the degree to which he can count on using it when he wants to. If it is located in close proximity to his other work, all the better.

It is thus believed that the only fair and workable arrangement to divide the large machine's capacity is to assure each large user a fixed minimum share of the large machine's

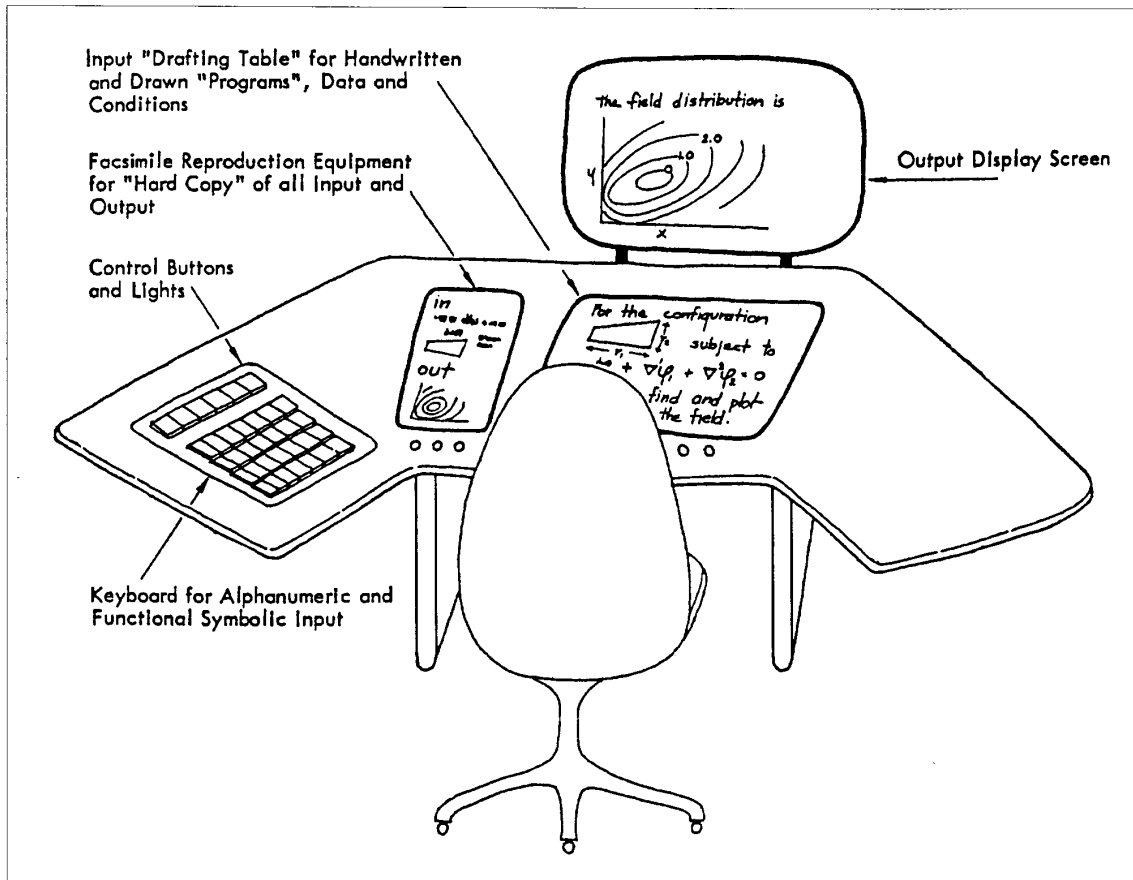


Figure 2. Teager's console. Reproduced from "M.I.T. Computation — Present and Future."⁴⁸

capacity, on call to him at any time he needs it, so that it has all the *properties* of a personal machine. If properly combined with remote facilities and languages, the user will be much happier with a percentage of a larger, and far more efficient data processor, than he would be with a much more limited capability machine of his very own. After all, as far as he is concerned, there would be no way of telling the difference, outside the fact that the time-shared facility would be far less costly to him, far easier to administrate, and much more powerful.

There is every indication that MIT will need at least a 20-100 fold increase in its available computer capacity within the next three years if it is to continue and accelerate its use of computers as an intellectual tool for research and teaching. The major alternatives for providing this capacity are the following:

1. Continue the present policy of allowing separate groups to provide their own separate facilities as present computer facilities become less and less able to meet their requirements.

2. Purchase or rent an extremely large central machine with adequate remote facilities to satisfy the projected needs and requirements of the various user groups.
3. Design or build our own large computer that would have the same or better capacity as would be provided by a machine that is available commercially, using presently existing components and machine organization techniques.
4. Designing or build our own large computer, using "state of the art" components, and the most advanced concepts of machine and system organization that can be obtained from the foremost experts in the field, whether they are at MIT, Lincoln, Rand, or elsewhere.

Of these major alternatives, buying a suitably modified commercial machine, and in particular Stretch, seems by far the most advantageous.

Time-Sharing at MIT

Minsky and Teager

Interestingly, in the workshop which formed the basis for the book *Computers and the World of the Future*²⁵ by Martin Greenberger, McCarthy says

The material I shall present on computer-system design was developed jointly with Marvin Minsky. I also want to acknowledge the stimulating effect of discussions with Professor Herbert M. Teager and Dr. F.J. Corbató, who are developing time-sharing systems for the IBM 7090 at the MIT Computation Center.

This contains one of only a few references to Marvin Minsky's contribution to the development of the concept; one other reference is in one of the translations of "MAC" provided by Peter Elias:

*Machine Aided Cognition
Man And Computer
Minsky Against Corbató*

Others have also provided *Multiple Access Computer*.

McCarthy also implies, and this was confirmed during the interviews, that Teager was proceeding with his own plans in parallel with Corbató. Teager's work was described in the *Communications of the ACM* in a 1962 reprint⁴⁹ of an earlier research report: "Real-Time, Time-Shared Computer Project," Computation Center and Research Laboratory of Electronics, MIT, Cambridge, Mass. Reported by Herbert M. Teager (Nov. 1961). Some excerpts follow (copyright 1962, Association for Computing Machinery, Inc., reprinted by permission):

The objective of this project is to develop devices, systems, and languages for

the fruitful interaction between scientists and computers, using the computer as a powerful, on-line aid to understanding. Due to cost and computer capacity considerations, this ability can best be provided within the context of time-sharing a slightly modified, standard memory size computer, equipped with random access files, among many low-cost simultaneously operating remote consoles, each equipped with low data rate graphical and character-producing input-output devices.

The major accomplishments of this project over the past calendar year are the following. On-line programming and computation utilizing a system of multiple independent typewriters has been tested. An existing digital plot-

Long Range Computation Study Group's recommendation for a time-sharing system.¹

The conclusions and recommendations of the Teager report did not reflect the views of the majority of the committee. Thus, a second report was submitted that recommended the acquisition of not just a large computing system, but also a system capable of supporting time-sharing.

The second report of the Long Range Computation Study Group was presented to Albert Hill in April 1961.

The major part of MIT's computational needs, a few years from now, can best be met by a single very high speed large-capacity computer system with provision for time-sharing through a number of remote consoles.

Present estimates of the cost of such a system lie between \$8 and \$25 million for a suitable central machine, including the development and procurement of remote input-output equipment, and the development of programming languages and systems appropriate for time-shared operation. However, this system would be the cheapest way of providing the needed capacity and the only way of providing the necessary, close man-machine interrelationship that we feel is vital for research in the years to come.

The committee is convinced that the bulk of scientific calculation is certain to increase rapidly and that it would be uneconomical, even if feasible, to try to keep pace with this growth by the addition of more and more standard machines, although there are areas in which separate machines may be valuable. Therefore, steps should be taken to acquire a giant machine meeting the specifications presented subsequently.

The conclusion that a central computer with remote consoles is required is based not only on the important economy obtained, both in running expenses and in programming expenses, but also on the importance of establishing close communication between the scientist and the machine. Thus, the committee feels that it is essential to provide for operating the machine in a time-sharing mode to provide immediate service to each of a large number of users simultaneously operating remote consoles throughout the Institute.

It is remarkable that, despite the many views and needs represented by members of the committee, there were essentially no compromises involved in coming to this general conclusion. Such a high-speed computing system having a very large memory and remote time-sharing operation meets the needs of very different kinds of projects, ranging from the numerical computations of physics, engineering and meteorology, to the symbolic computations of language translation and artificial intelligence projects.

ter has been connected to the IBM 709 computer, and a special-purpose computer to control multiple independent plotters is being constructed. A prototype of a high resolution graphical input device for hand-drawn figures and symbols is being built. Design modifications to the 7090 computer have been proposed and are being incorporated in our forthcoming machine. Scheduling systems for time-sharing and memory allocation have been simulated and found satisfactory. An information retrieval system for programs and data is being designed. Programming systems to allow handwriting as a valid input are being checked out, and new graphical languages for several major problem classes for input and output have been partially specified.

REFERENCES:

Teager, H. Dec. 1960. Marriage of on-line human decision with computer programs. *Naval Res. Logist. Quart.*, 379-383.

As indicated in the interviews later, Teager apparently concentrated on the design of interactive input-output devices connected to the IBM 709 at the same time as Corbató was implementing CTSS on the same machine.

In the above communication, Teager refers to a prior paper presented at George Washington University in 1960, which appears to provide the motivation for his work. This motivation, typified here by the abstract from that paper, has a strong resemblance to Licklider's man-computer symbiosis concept.

As the logistics problems of current interest become more and more complex, it becomes apparent that heuristic methods hold more promise than exhaustive or algorithmic methods of

solution. It is also becoming clearer that heuristics, per se, are useless in approaching large combinatorial problems (via pattern recognition, for example), unless some general techniques are developed to draw out the specific "best" heuristics required for each problem. It is a fact that humans are, at present, far more efficient in developing and applying such heuristics than a machine. It is therefore of considerable value to explore how the best features of both human and machine decision can be married. The general problem requires the development of common meaningful languages, specialized input-output display and entry facilities, and a means of time-sharing the computer, in order to efficiently match the machine versus the human-decision time.⁴⁷

Feasibility of proposed system

All components of the proposed central computer system either exist or can be developed during the time necessary for the procurement of the central computer. For example, time-sharing arrangements using remote input/output stations equipped with typewriters presently exist. Further study is needed in the areas of console specifications, programming languages and system administration, but the Study Group is confident that the present recommendations are not dependent upon the results of these studies. In defining these areas, it is merely recognizing the need to present a full picture of future work.

The final choice for the central computer should be based on whether it can be installed in three years and will meet the set of general specifications [presented in Part III of this report]. There are two feasible alternate ways to acquire such a computer. The first is to purchase or rent one of the very large commercially available computers, such as the IBM 7030, commonly known as "Stretch." With minor modifications and provisions of time-sharing facilities, this computer will meet many of the requirements. The second alternative is to have a manufacturer construct a computer according to our specifications, making full use of any advanced components which he may have.

It is not possible at this time to make any kind of firm cost estimate, since the committee has not yet been in direct contact with manufacturers. A suitable system based on the IBM Stretch computer would, at rumored prices, cost between \$20,000,000 and \$25,000,000. Much of this cost would be the cost of memory. Some of the members of the committee believe that the cost of a suitable system may be as little as seven or eight million, but others are not convinced.

The importance of time-sharing

The main conclusion of this report is not just that MIT acquire a very large computer system, but that the computer system be a time-sharing one, accessible to its users through remote consoles and accessible to laboratory apparatus through suitable data transmission facilities. The reason for this emphasis on time-sharing is because the greatest shortage at MIT is not in computer speed but in interaction capability with the users and with the users' laboratories. In fact, the increase in interaction capability made possible by the time-shared system will have as much impact on MIT research as the introduction of automatic computation in the first place.

A single, very large, time-shared system is recommended for the following reasons:

Time-Sharing at MIT

1. Only a very large computer can satisfy the needs of users with very large problems.
2. Only a time-sharing system can make possible efficient use of large capacity by many small users.
3. Only a very large secondary storage system can contain all programs and data and make them rapidly available to the computer under the control of simple consoles.
4. Only a single large system with many users can provide a sufficient base to support the large effort required to provide advanced public programs to all users on an adequate scale.
5. Only a time-sharing system can provide the interaction capability which, especially when combined with advanced programming and control languages, is so essential to improved computer utilization in research.

Specifications for central computer

The general scale and characteristics of the central computer are indicated by the following specifications:

1. An advanced indexing and addressing system, fixed and floating point arithmetic, and a full complement of logical and variable length operation.
2. Average program running speed of at least 10^6 instructions per second.
3. 250,000 to 1,000,000 words of directly addressable memory, each word at least 48 bits.
4. Full time-sharing facilities including memory protection, interrupt capability, and non-stop operation.
5. Provision for a system of remote consoles.
6. Capability for real-time interaction with laboratory apparatus at a rate of 250,000 words/sec.
7. Secondary storage of more than 10^7 words with an access time of about .1 sec. and a data rate of 250,000 words/sec. ■

References and Bibliography

1. D.N. Arden et al., "Report of the Long Range Computation Study Group," unpublished report, MIT, Cambridge, Mass., 1961. This is the second report on this topic from the MIT study group, developed by the members independent of the chairman (Herbert Teager). The report is addressed to A.G. Hill.
2. M.M. Astrahan and J.F. Jacobs, "History of the Design of the SAGE Computer — The AN/FSQ-7," *Annals of the History of Computing*, Vol. 5, No. 4, 1983, pp. 340-349. While this is a broad-brush overview of the design of the SAGE system, the authors claim a number of innovations for the system including: "Time-sharing. Time-sharing a computer for real-time tracking of hundreds of airplanes, real-time control of weapons, and interaction with human controllers was a bold concept. It required invention of programming techniques to ensure timely sequencing through all the tasks. Programs and data tables were paged in from drums, and only the tables were rewritten. Data input-output and display data were fully buffered by drums."
3. W.F. Bauer, "Computer Design from the Programmer's Viewpoint," *Proc. Eastern Joint Computer Conf.*, AFIPS, Reston, Va., 1959, p. 49. The exposition of the concept of a large-scale computer as a "utility."
4. R.W. Bemer, "How to Consider a Computer," *Automatic Control*, Mar. 1957, pp. 66-69.
5. R.W. Bemer, "PRINT 1 — An Automatic Coding System for the IBM 705," *J. Franklin Inst. Series*, Mono. No. 3, Apr., 1957. See Discussion, p. 38. A claimed first use of the word "time-sharing."
6. R.W. Bemer, Letter to the Editor, *Scientific American*, Jan. 1967, p. 8. Claims that his article in the 1957 issue of *Automatic Control* is an original reference to time-sharing.
7. R.W. Bemer, Letter to George R. Trimble, 1968. Correction to bibliography on time-sharing that appeared in *Computing Reviews*, Vol. 14, May 1968, pp. 291-301.
8. V. Bush "As We May Think," *Atlantic Monthly*, Vol. 176, July 1945, pp. 101-108.
9. M. Campbell-Kelly, "Christopher Strachey 1916-1975: A Biographical Note," *Annals of the History of Computing*, Vol. 7, No. 1, 1985, pp. 19-42. Contains a brief overview of Strachey's involvement in "multiaccess time-sharing computers."
10. P. Ceruzzi, "When Computers Were Human," *Annals of the History of Computing*, Vol. 13, No. 3, 1991, pp. 237-244.
11. S.A. Coons, "An Outline of the Requirements for a Computer-Aided Design System," *Proc. Spring Joint Computer Conf.*, Vol. 23, AFIPS, Spartan Books, Washington, D.C., 1963, pp. 299-304. Describes the broad concept of computer-aided design; one of the early applications of the MIT Cooperative Computer Laboratory. Contains references to Douglas Ross's early work on APT.
12. F.J. Corbató et al., *The Compatible Time-Sharing System: A Programmer's Guide*, MIT Press, Cambridge, Mass., 1963.
13. F.J. Corbató, "Time-sharing" in *Encyclopedia of Computer Science and Engineering*, 2nd edition, A. Ralston and E.D. Reilly, Jr., eds., Van Nostrand Reinhold, New York, 1983, pp. 1520-1526.
14. F.J. Corbató, M. Merwin-Daggett, and R.C. Daley, "An Experimental Time-Sharing System," *Proc. Spring Joint Computer Conf.*, Vol. 21, AFIPS, National Press, Palo Alto, Calif., 1962, pp. 335-344. Perhaps the first publication on CTSS which not only gave performance figures for the system but also suggested that such a system could be developed for other "contemporary computers." Contains a short bibliography on prior publications related to the concept of time-sharing.
15. G.J. Culler, "A User's View of a Functionally Oriented On-Line System," *Proc. IBM Scientific Computing Symp. Man-Machine Communication*, IBM Data Processing Division, White Plains, N.Y., 1966, p. 21 [abstract only].
16. R.C. Daley and P.G. Neumann, "A General Purpose File System for Secondary Storage," *Proc. Fall Joint Computer Conf.*, Vol. 27, AFIPS, Spartan Books, Washington, D.C., 1965, pp. 213-229.
17. R.W. DeSio, ed., *Proc. IBM Scientific Computing Symp. Man-Machine Communication*, IBM Data Processing Division, White Plains, N.Y., 1966. A symposium that contains several papers on the design and implementation of time-sharing systems as well as users' experiences. See other entries for individual papers by Taub, Culler, Fano, and Shaw.
18. R.M. Dodds, Letter to R.W. Bemer, 1967. Dodds claims invention of time-sharing (although not under that title) in a 1949 letter to IBM.
19. B.O. Evans, Part of the Discussion of the SPREAD Report, *Annals of the History of Computing*, Vol. 5, No. 1, 1983, pp. 34-35. In a response to Aaron Finerman's suggestion that "IBM shied away from the interactive approach." Evans explains the

- beginning of the development of the System 360/67 and TSO (Time-Sharing Option).
20. R.R. Everett, C.A. Zraket, and H.D. Bennington, "SAGE — A Data Processing System for Air Defense," *Proc. EJCC*, IRE (now IEEE), Washington, D.C., 1957, pp. 148-155. Reprinted in *Annals of the History of Computing*, Vol. 5, No. 4, 1983, pp. 330-339. A description of the SAGE system (AN/FSQ-7). It includes a treatment of the message buffering procedures utilizing drum storage and the need to restrict the processing time in response to the messages to a minimum since the time of receipt of messages is "often critical." Later in this description, the authors use the word "time-sharing" without any prior (or subsequent) definition.
 21. R.M. Fano and F.J. Corbató, "The Time-Sharing of Computers," *Scientific American*, Sept. 1966, pp. 128-140.
 22. R.M. Fano, and F.J. Corbató, Letter to the Editor, *Scientific American*, Jan. 1967, p. 8. Statement of the contribution of John McCarthy and others, besides Chris Strachey, to the development of time-sharing.
 23. K.A. Frenkel, "Allan L. Scherr: Big Blue's Time-Sharing Pioneer," *Profiles in Computing*, *Comm. ACM*, Vol. 30, No. 10, Oct. 1987, pp. 824-828. A profile of Allan L. Scherr on the occasion of the 40th anniversary of ACM — alongside Don Knuth and Brian K. Reid, two other winners of the ACM Grace Murray Hopper Award.
 24. E.L. Glaser et al., "System Design of a Computer for Time-sharing Applications," *Proc. Fall Joint Computer Conf.*, Vol. 27, AFIPS, Spartan Books, Washington, D.C., 1965, pp. 197-202. The design of the GE 645 computer for use with the Multics system.
 25. M. Greenberger, *Computers and the World of the Future*, MIT Press, Cambridge, Mass., 1962.
 26. M. Greenberger, "Computers of Tomorrow," *Atlantic Monthly*, May 1964.
 27. S. Levy, *Hackers: Heroes of the Computer Revolution*, Anchor Press/Doubleday, Garden City, N.Y., 1984. An excellent book on the phenomenon of hacking from its semirespectable days to the mid-1980s, when it was disreputable.
 28. J.C.R. Licklider, "Man-Computer Symbiosis," *IRE Trans. Human Factors in Electronics*, Mar. 1960, pp. 4-11.
 29. W.F. Luebbert, "Commemoration of 1940 Remote Computing Demonstration by Stibitz," *Annals of the History of Computing*, Vol. 3, No. 1, 1981, pp. 68-70. An account of the placement of a plaque at Dartmouth College commemorating this first demonstration.
 30. M.J. Marcotty, F.M. Longstaff, and A.P.M. Williams, "Time-Sharing on the Ferranti-Packard FP6000 Computer System," *Proc. Spring Joint Computer Conf.*, Vol. 23, AFIPS, Spartan Books, Washington, D.C., 1963, pp. 29-40. An example of a noninteractive (nonterminal) system in which time-sharing optimized the use of a processor connected to peripherals of different speeds.
 31. S.L. Marks, "JOSS — Conversational Computing for the Non-programmer," *Annals of the History of Computing*, Vol. 4, No. 1, 1982, pp. 35-52. From the abstract: "JOSS is a conversational time-sharing system developed at the Rand Corporation to demonstrate, on a small scale, the value of time-sharing and easy access to computing power for the nonprogrammer."
 32. A. Mastrogiovanni, "Operating Systems: Some Reflections After a Meeting," *Annals of the History of Computing*, Vol. 10, No. 3, 1988, pp. 195-209.
 33. J. McCarthy, "A Time-sharing Operator Program for Our Projected IBM 709," unpublished memorandum, MIT, Cambridge, Mass., 1959. The original memorandum, addressed to Philip M. Morse, outlining the design of a time-sharing system.
 34. J. McCarthy, "Time-Sharing Computer Systems," in *Computers and the World of the Future*, M. Greenberger, ed., MIT Press, Cambridge, Mass., 1962, pp. 221-248.
 35. J. McCarthy, S. Boilen, E. Fredkin, and J.C.R. Licklider, "A Time-Sharing Debugging System for a Small Computer," *Proc. Spring Joint Computer Conf.*, Vol. 23, AFIPS, Spartan Books, Washington, D.C., 1963, pp. 51-57. A description of the BBN time-sharing system for the PDP-1.
 36. R.F. Rosin, "Query on TSS," Letter to the Editor and Response by Bernard A. Galler, *Annals of the History of Computing*, Vol. 5, No. 4, 1983, p. 404. A short discussion of the development of TSS through the cooperative efforts of IBM, the University of Michigan, and others.
 37. D.T. Ross, "Design of Special Language for Machine-Tool Programming," in *MIT Servo Lab Course Outline and Workbook for the Special Course on Programming for Numerically Controlled Machine Tools*, MIT, Cambridge, Mass., 1957.
 38. D.T. Ross and J.E. Rodriguez, "Theoretical Foundations for the Computer-Aided Design System," *Proc. Spring Joint Computer Conf.*, Vol. 23, AFIPS, Spartan Books, Washington, D.C., 1963, pp. 305-322. A companion to the paper of a similar title by S.A. Coons. It discusses AED language processing, system structure, and symbolic computation solutions to servomechanism design, as an example. AED-0 language was the first machine-independent software engineering language. This paper won a prize as the best paper of the conference.
 39. A.L. Scherr, "Time-sharing Measurement," *Datamation*, Apr. 1966.
 40. J.I. Schwartz et al., "A General Purpose Time-Sharing System," *Proc. Spring Joint Computer Conf.*, Vol. 25, AFIPS, Spartan Books, Washington, D.C., 1964, pp. 397-411. A description of the JOSS system of timesharing.
 41. J.C. Shaw, "JOSS: Experience with an Experimental Computing Service for Users at Remote Typewriter Consoles," *Proc. IBM Scientific Computing Symp. Man-Machine Communication*, IBM Data Processing Division, White Plains, N.Y., 1966, pp. 23-32.
 42. R. Stotz, "Man-Machine Console Facilities for Computer-Aided Design," *Proc. Spring Joint Computer Conf.*, Vol. 23, AFIPS, Spartan Books, Washington, D.C., 1963, pp. 323-328. A third paper in a sequence of articles on computer-aided

Time-Sharing at MIT

design. No mention of CTSS, but it mentions MIT computing facilities.

43. C. Strachey, "Time-sharing in Large Fast Computers," *Proc. Int'l Conf. Information Processing, UNESCO, Paris 15-20 June 1959*, UNESCO, Paris, 1960, pp. 336-341. Generally thought of as the first "published" account of time-sharing, but in some ways Strachey's concept of time-sharing was different from that of John McCarthy, whose ideas were contemporaneous.
44. C. Strachey, Letter to Don Knuth, May 1, 1974. "The paper I wrote called 'Time-sharing in Large Fast Computers' was read at the first (pre IFIP) conference at Paris in 1960. It was mainly about multi-programming (to avoid waiting for peripherals) although it did envisage this going on at the same time as a programmer was debugging his program at a console. I did not envisage the sort of console system which is now so confusingly called time-sharing. I still think my use of the term is the more natural."
45. A.H. Taub, "On Time-Sharing Systems, Design and Use," *Proc. IBM Scientific Computing Symp. Man-Machine Communication*, IBM Data Processing Division, White Plains, N.Y., 1966, pp. 9-16.
46. R.W. Taylor, "In Memoriam: J.C.R. Licklider 1915-1990," Research Report 61, System Research Center, Digital Equipment Corp., Palo Alto, Calif., 1990, with additional notes by Robert Rosin.
47. H.M. Teager, "The Marriage of On-Line Human Decision with Computer Programs," *Naval Research Logistics Quarterly*, Vol. 7, No. 4, 1960, pp. 379-383. An extended abstract of a presentation at the Decennial Logistics Research Conference which expresses some of the same thoughts as the paper by Licklider on man-computer symbiosis. Briefly discusses the features required of a computer to support time-sharing.
48. H.M. Teager, "M.I.T. Computation — Present and Future," MIT unpublished report, Cambridge, Mass., 1961. This is the first report of the Long Range Computation Study Group generated under the chairmanship of Teager. The group was unable to support the conclusion to acquire an IBM 7030 (Stretch). Thus a second report, developed under the chairmanship of John McCarthy, recommended the approach used in CTSS.
49. H.M. Teager, "Real-Time, Time-Shared Computer Project," Research Review, *Comm. ACM*, Vol. 5, No. 1, 1962, p. 62. A brief description dated November 1961 "to develop devices, systems and languages for the fruitful interaction between scientists and computers, using the computer as a powerful, on-line aid to understanding."
50. G.R. Trimble, "A Time-sharing Bibliography," Bibliography 14, *ACM Computing Reviews*, May 1968, pp. 291-301.
51. C.R. Wieser, "The Cape Cod System," *Annals of the History of Computing*, Vol. 5, No. 4, Oct. 1983, pp. 362-369.
52. K.L. Wildes and N.A. Lingren, "The Evolution of Time-sharing," in *A Century of Electrical Engineering and Computer Science at MIT, 1882-1982*, MIT Press, Cambridge, Mass., 1986.
53. M. Wilkes, *Memoirs of a Computer Pioneer*, MIT Press, Cambridge, Mass., 1985.